Conformant Planning as a Case Study of Incremental QBF Solving

Uwe Egly, Martin Kronegger, Florian Lonsing, Andreas Pfandler

Institute of Information Systems
Vienna University of Technology, Austria
firstname.lastname@tuwien.ac.at

12th International Conference on Artificial Intelligence and Symbolic Computation, December 11 - 13, 2014, Sevilla, Spain

This work is supported by the Austrian Science Fund (FWF) under grants S11409-N23 and P25518-N23 and by the German Research Foundation (DFG) under grant ER 738/2-1.

Overview (1/2)

Quantified Boolean Formulas (QBF):

- Propositional formulae with universally ($\forall$) and ($\exists$) existentially quantified propositional variables.
  E.g. $\exists x \forall y \exists z.\ C_1 \wedge C_2 \wedge \ldots \wedge C_n$.
- Solving a QBF: PSPACE-complete.
- Applications in model checking, formal verification, testing,...
- Our focus: conformant planning (Σ_2^P-complete).

Overview (1/2)

Quantified Boolean Formulas (QBF):

- Propositional formulae with universally ($\forall$) and ($\exists$) existentially quantified propositional variables.
  E.g. $\exists x \forall y \exists z.\ C_1 \wedge C_2 \wedge \ldots \wedge C_n$.
- Solving a QBF: PSPACE-complete.
- Applications in model checking, formal verification, testing,...
- Our focus: conformant planning (Σ_2^P-complete).

QBF in Practice:

- In practice, often a sequence $\psi_0, \psi_1, \ldots, \psi_n$ of related formulas must be solved.
- Try to exploit similarity between formulas in a sequence.
- Information gathered when solving ψ_i might help to solve ψ_j with $j > i$.
- A non-incremental solver forgets everything learned from ψ_i when solving ψ_j.

Overview (2/2)

QBF-Based Approach to Conformant Planning:

- Generic QBF-based workflow to solve conformant planning problems.
- Precision: we always find the optimal solution (given sufficient time and memory). This is in contrast to heuristic approaches.
- Workflow implemented in a Java tool.
- Our focus: comparison of incremental and non-incremental QBF solving by DepQBF.
- Experiments: incremental use of DepQBF performs best.

Overview (2/2)

QBF-Based Approach to Conformant Planning:

- Generic QBF-based workflow to solve conformant planning problems.
- Precision: we always find the optimal solution (given sufficient time and memory). This is in contrast to heuristic approaches.
- Workflow implemented in a Java tool.
- Our focus: comparison of incremental and non-incremental QBF solving by DepQBF.
- Experiments: incremental use of DepQBF performs best.

DepQBF:

- General purpose, award-winning incremental QBF solver.
- Free software: `http://lonsing.github.io/depqbf/`
- Related work:
  - Lonsing, Egly: Incremental QBF Solving. In Proc. CP 2014.

Conformant Planning

- Variant of classical AI planning.
- Given: initial state s_0 defined as a set of variables with some unknown values.
- Given: a set of (nondeterministic) actions with preconditions and effects. An action, when executed on a state s under preconditions, produces a successor state s'.
- Given: a set of goal states.
- Find a sequence of actions (called a *plan*) from the initial state s_0 to a goal state which works out with respect to all possible values of the unknown variables.
- QBF-based approach: encodings by Rintanen (2007); can be competitive to other planning tools (c.f. Kronegger, Pfandler, Pichler (2013).

Conformant Planning

- Variant of classical AI planning.
- Given: initial state s_0 defined as a set of variables with some unknown values.
- Given: a set of (nondeterministic) actions with preconditions and effects. An action, when executed on a state s under preconditions, produces a successor state s'.
- Given: a set of goal states.
- Find a sequence of actions (called a *plan*) from the initial state s_0 to a goal state which works out with respect to all possible values of the unknown variables.
- QBF-based approach: encodings by Rintanen (2007); can be competitive to other planning tools (c.f. Kronegger, Pfandler, Pichler (2013).

Our Benchmark Problem: "Dungeon"

- Captures full hardness of conformant planning (Σ_2^P-complete).
- Search for plan can be encoded as a QBF with prefix $\exists\forall$.

- Given: a dungeon with monsters a player wants to fight.
- Player needs certain items to defeat a particular monster.
- Before entering the dungeon, player can pick items from pools.
- Unknown: hidden pools of special items the player picks.
- Goal: defeat all monsters in the dungeon regardless of special items the player gets.
- Parameters: number of monsters, pools, items necessary to defeat a monster,. . .

Dungeon Benchmark

- Given: a dungeon with monsters a player wants to fight.
- Player needs certain items to defeat a particular monster.
- Before entering the dungeon, player can pick items from pools.
- Unknown: hidden pools of special items the player picks.
- Goal: defeat all monsters in the dungeon regardless of special items the player gets.
- Parameters: number of monsters, pools, items necessary to defeat a monster,. . .

Our Approach:

- Encode the search for a plan as QBFs to be solved by a QBF solver.

QBF-Based Approach to Conformant Planning

- Idea: search for plans of increasing lengths $i = 0, 1, \ldots$.
- QBF encoding ψ_i parametrized by plan length i: $\psi_0, \psi_1, \ldots$.
- There exists a plan of length i if the QBF ψ_i is satisfiable.
- If ψ_i is unsatisfiable, then set $i := i + 1$ and tackle ψ_{i+1}.

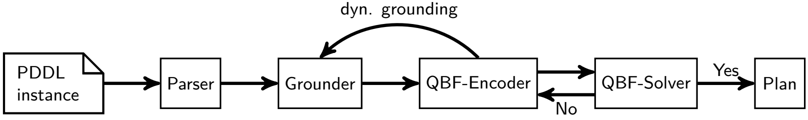
QBF-Based Approach to Conformant Planning

- Idea: search for plans of increasing lengths $i = 0, 1, \ldots$.
- QBF encoding ψ_i parametrized by plan length i: $\psi_0, \psi_1, \ldots$.
- There exists a plan of length i if the QBF ψ_i is satisfiable.
- If ψ_i is unsatisfiable, then set $i := i + 1$ and tackle ψ_{i+1}.

Our Approach:

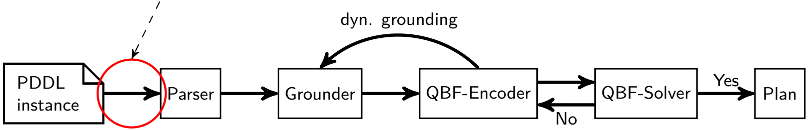
- QBF-based workflow including encoding and solving implemented in a Java tool.
- Our tool is generic and can solve arbitrary conformant planning problems.
- QBF-based approach to conformant planning always finds the optimal (shortest) plan, in contrast to heuristic search for plan.

QBF-Based Approach: Non-Incremental QBF Solving



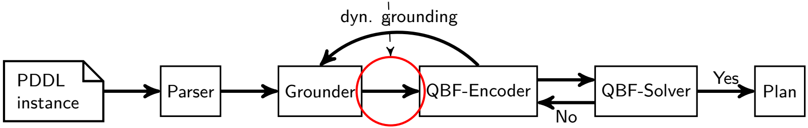
QBF-Based Approach: Non-Incremental QBF Solving

```
(:init (inPool i1 pool1)
       (inPool i4 pool1)
       (inPool i2 pool2)
       (unknown (in u1)) ···
)
(:goal (win g) )
(:action pick
  :parameters (?i - item ?p - pool)
  :precondition (and (allowPick ?p) (inPool ?i ?p))
  :effect (and (in ?i) (not (allowPick ?p)))
)
(:action fight ···
  :effect (and
          (when (and (in i2) (not (in u3)) (in s1)) (win g))
          (when (and (not (in i4)) (in u1)) (win g))
```



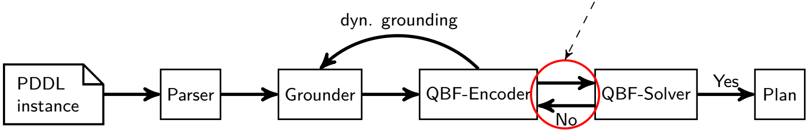
QBF-Based Approach: Non-Incremental QBF Solving

```
(:action pick
  :precondition (and (allowPick pool1) (inPool i1 pool1))
  :effect (and (in i1) (not (allowPick pool1)))
)
(:action pick
  :precondition (and (allowPick pool2) (inPool i1 pool2))
  :effect (and (in i1) (not (allowPick pool2)))
)
...
(:action pick
  :precondition (and (allowPick pool1) (inPool i2 pool1))
  :effect (and (in i2) (not (allowPick pool1)))
)
...
```



QBF-Based Approach: Non-Incremental QBF Solving

```
p cnf 1666 4833
e 516 518 512 514 515 1024 1025 1026 1028 1536 ... 0
a 34 118 101 130 46 ... 0
e 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 ... 0
1 0
...
-30 0
...
-139 140 0
-139 -141 0
-177 46 34 130 -118 178 0
-101 193 0
-348 189 190 191 -192 349 0
-348 152 -193 349 0
...
```



A Closer Look at the QBF Encoding

- Without Optimization, based on Rintanen [2007].
- Goal: A quantified Boolean formula ψ that is true iff the corresponding planning problem has a plan of length k.

A Closer Look at the QBF Encoding

- Without Optimization, based on Rintanen [2007].
- Goal: A quantified Boolean formula ψ that is true iff the corresponding planning problem has a plan of length k.

$$\psi := \exists \text{ actions } \forall \text{ unknowns } \exists \text{ helpers } \varphi$$

A Closer Look at the QBF Encoding

- Without Optimization, based on Rintanen [2007].
- Goal: A quantified Boolean formula ψ that is true iff the corresponding planning problem has a plan of length k.

$$\psi := \exists \text{ actions } \forall \text{ unknowns } \exists \text{ helpers } \varphi$$

$$\varphi := \underbrace{\bigwedge_{v \in \mathcal{PI}} v^0 \wedge \bigwedge_{v \in \mathcal{NI}} \neg v^0}_{\text{initial state}} \wedge \underbrace{\bigwedge_{v \in \mathcal{PG}} v^k \wedge \bigwedge_{v \in \mathcal{NG}} \neg v^k}_{\text{goal}}$$

A Closer Look at the QBF Encoding

- Without Optimization, based on Rintanen [2007].
- Goal: A quantified Boolean formula ψ that is true iff the corresponding planning problem has a plan of length k.

$$\psi := \exists \text{ actions } \forall \text{ unknowns } \exists \text{ helpers } \varphi$$

$$\varphi := \underbrace{\bigwedge_{v \in \mathcal{PI}} v^0 \wedge \bigwedge_{v \in \mathcal{NI}} \neg v^0}_{\text{initial state}} \wedge \underbrace{\bigwedge_{v \in \mathcal{PG}} v^k \wedge \bigwedge_{v \in \mathcal{NG}} \neg v^k}_{\text{goal}}$$

$$\wedge \bigwedge\nolimits_{t \in [k]} \bigwedge\nolimits_{a \in \mathcal{A}^{t-1}} \left[\left(a^{t-1} \wedge \bigwedge\nolimits_{p \in \mathrm{pre}(a)} p^{t-1} \right) \rightarrow \bigwedge\nolimits_{e \in \mathrm{eff}(a)} e^t \right]$$

A Closer Look at the QBF Encoding

- Without Optimization, based on Rintanen [2007].
- Goal: A quantified Boolean formula ψ that is true iff the corresponding planning problem has a plan of length k.

$$\psi := \exists \text{ actions } \forall \text{ unknowns } \exists \text{ helpers } \varphi$$

$$\varphi := \underbrace{\bigwedge_{v \in \mathcal{PI}} v^0 \wedge \bigwedge_{v \in \mathcal{NI}} \neg v^0}_{\text{initial state}} \wedge \underbrace{\bigwedge_{v \in \mathcal{PG}} v^k \wedge \bigwedge_{v \in \mathcal{NG}} \neg v^k}_{\text{goal}}$$

$$\wedge \bigwedge_{t \in [k]} \bigwedge_{a \in \mathcal{A}^{t-1}} \left[\left(a^{t-1} \wedge \bigwedge_{p \in \mathrm{pre}(a)} p^{t-1} \right) \rightarrow \bigwedge_{e \in \mathrm{eff}(a)} e^t \right]$$

$\wedge$ one action at a time

A Closer Look at the QBF Encoding

- Without Optimization, based on Rintanen [2007].
- Goal: A quantified Boolean formula ψ that is true iff the corresponding planning problem has a plan of length k.

$$\psi := \exists \text{ actions } \forall \text{ unknowns } \exists \text{ helpers } \varphi$$

$$\varphi := \underbrace{\bigwedge_{v \in \mathcal{PI}} v^0 \wedge \bigwedge_{v \in \mathcal{NI}} \neg v^0}_{\text{initial state}} \wedge \underbrace{\bigwedge_{v \in \mathcal{PG}} v^k \wedge \bigwedge_{v \in \mathcal{NG}} \neg v^k}_{\text{goal}}$$

$$\wedge \bigwedge_{t \in [k]} \bigwedge_{a \in \mathcal{A}^{t-1}} \left[\left(a^{t-1} \wedge \bigwedge_{p \in \mathrm{pre}(a)} p^{t-1} \right) \rightarrow \bigwedge_{e \in \mathrm{eff}(a)} e^t \right]$$

$\wedge$ one action at a time

$\wedge$ every change has a cause (framing axioms)

A Closer Look at the QBF Encoding

- Without Optimization, based on Rintanen [2007].
- Goal: A quantified Boolean formula ψ that is true iff the corresponding planning problem has a plan of length k.

$$\psi := \exists \text{ actions } \forall \text{ unknowns } \exists \text{ helpers } \varphi$$

$$\varphi := \underbrace{\bigwedge_{v \in \mathcal{PI}} v^0 \wedge \bigwedge_{v \in \mathcal{NI}} \neg v^0}_{\text{initial state}} \wedge \underbrace{\bigwedge_{v \in \mathcal{PG}} v^k \wedge \bigwedge_{v \in \mathcal{NG}} \neg v^k}_{\text{goal}}$$

$$\wedge \; \bigwedge_{t \in [k]} \bigwedge_{a \in \mathcal{A}^{t-1}} \left[\left(a^{t-1} \wedge \bigwedge_{p \in \mathrm{pre}(a)} p^{t-1} \right) \rightarrow \bigwedge_{e \in \mathrm{eff}(a)} e^t \right]$$

$\wedge$ one action at a time

$\wedge$ every change has a cause (framing axioms)

A Closer Look at the QBF Encoding

- Without Optimization, based on Rintanen [2007].
- Goal: A quantified Boolean formula ψ that is true iff the corresponding planning problem has a plan of length k.

$$\psi := \exists \text{ actions } \forall \text{ unknowns } \exists \text{ helpers } \varphi$$

$$\varphi := \underbrace{\bigwedge_{v \in \mathcal{PI}} v^0 \wedge \bigwedge_{v \in \mathcal{NI}} \neg v^0}_{\text{initial state}} \wedge \underbrace{\bigwedge_{v \in \mathcal{PG}} v^k \wedge \bigwedge_{v \in \mathcal{NG}} \neg v^k}_{\text{goal}}$$

$$\wedge \bigwedge_{t \in [k]} \bigwedge_{a \in \mathcal{A}^{t-1}} \left[\left(a^{t-1} \wedge \bigwedge_{p \in \mathrm{pre}(a)} p^{t-1} \right) \rightarrow \bigwedge_{e \in \mathrm{eff}(a)} e^t \right]$$

$\wedge$ one action at a time

$\wedge$ every change has a cause (framing axioms)

A Closer Look at the QBF Encoding

- Without Optimization, based on Rintanen [2007].
- Goal: A quantified Boolean formula ψ that is true iff the corresponding planning problem has a plan of length k.

$$\psi := \exists \text{ actions } \forall \text{ unknowns } \exists \text{ helpers } \varphi$$

$$\varphi := \underbrace{\bigwedge_{v \in \mathcal{PI}} v^0 \wedge \bigwedge_{v \in \mathcal{NI}} \neg v^0}_{\text{initial state}} \wedge \underbrace{\bigwedge_{v \in \mathcal{PG}} v^k \wedge \bigwedge_{v \in \mathcal{NG}} \neg v^k}_{\text{goal}}$$

$$\wedge \bigwedge_{t \in [k]} \bigwedge_{a \in \mathcal{A}^{t-1}} \left[\left(a^{t-1} \wedge \bigwedge_{p \in \mathrm{pre}(a)} p^{t-1} \right) \to \bigwedge_{e \in \mathrm{eff}(a)} e^t \right]$$

$\wedge$ one action at a time

$\wedge$ every change has a cause (framing axioms)

Stacks of Clauses in the QBFs for Plan Lengths i and $i + 1$

goal(i)
trans($i - 1$,i)
$\vdots$
trans(1,2)
trans(0,1)
init

Stacks of Clauses in the QBFs for Plan Lengths i and $i+1$

goal(i)	goal(i)
trans($i-1$,i)	trans($i-1$,i)
$\vdots$	$\vdots$
trans(1,2)	trans(1,2)
trans(0,1)	trans(0,1)
init	init

Stacks of Clauses in the QBFs for Plan Lengths i and $i+1$

goal(i)	
trans($i-1$,i)	trans($i-1$,i)
$\vdots$	$\vdots$
trans(1,2)	trans(1,2)
trans(0,1)	trans(0,1)
init	init

Stacks of Clauses in the QBFs for Plan Lengths i and $i + 1$

goal(i)	trans(i,$i + 1$)
trans($i - 1$,i)	trans($i - 1$,i)
$\vdots$	$\vdots$
trans(1,2)	trans(1,2)
trans(0,1)	trans(0,1)
init	init

Stacks of Clauses in the QBFs for Plan Lengths i and $i+1$

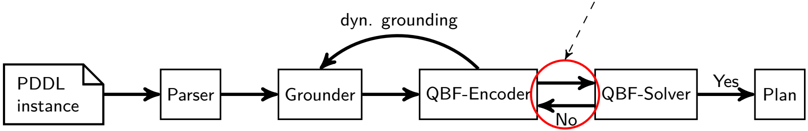
goal(i)
trans($i-1$,i)
$\vdots$
trans(1,2)
trans(0,1)
init

goal($i+1$)
trans(i,$i+1$)
trans($i-1$,i)
$\vdots$
trans(1,2)
trans(0,1)
init

- QBFs ψ_i and ψ_{i+1} encoding plan lengths i and $i+1$ share many clauses.
- Use an incremental QBF solver to exploit similarity between ψ_i and ψ_{i+1}.

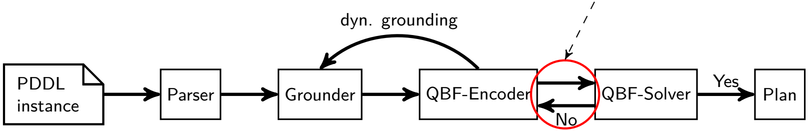
QBF-Based Approach: Incremental QBF Solving by DepQBF

```
// assume u ∈ ℕ is a lower bound on the plan length

DepQBF4J.create();
DepQBF4J.configure("–dep-man=simple");
DepQBF4J.configure("–incremental-use");

DepQBF4J.newScopeAtNesting(DepQBF4J.QTYPE_EXISTS,1);
DepQBF4J.add(0);
// add other quantifiers

// add clauses for initial state
// e.g.: DepQBF4J.add(1); DepQBF4J.add(0);
// add clauses for trans(j − 1,j)    ∀j ∈ {1, . . . , u}

DepQBF4J.push(); // add new frame

// add clauses for goal(u)

int ret = DepQBF4J.sat();
```



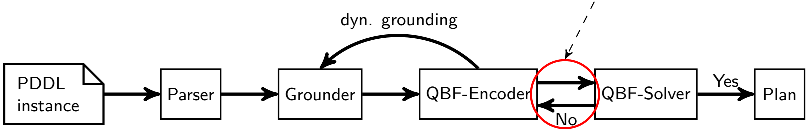
QBF-Based Approach: Incremental QBF Solving by DepQBF

// assume the QBF for plan length i is UNSAT



QBF-Based Approach: Incremental QBF Solving by DepQBF

```
// assume the QBF for plan length i is UNSAT

DepQBF4J.reset(); // reset solver state but keep learned information
DepQBF4J.pop(); // delete old goal clauses (remove top frame)

// add clauses for trans(i,i + 1)

DepQBF4J.push(); // add new frame

// add clauses for goal(i + 1)

int ret = DepQBF4J.sat();
```



Experiments (1/3): Overall Statistics

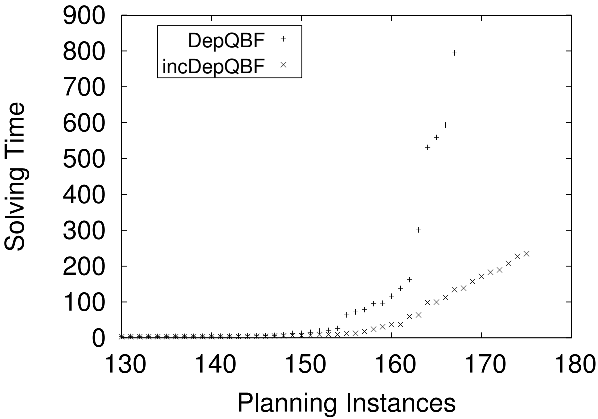
Goal of the experimental evaluation

Compare incremental and non-incremental QBF solving for conformant planning.

288 Planning Instances (Dungeons Benchmark)							
	Time	Solved	Plan found	No plan	$\overline{t}$	$\overline{b}$	$\overline{a}$
DepQBF:	112,117	168	163	5	24.40	2210	501,706
incDepQBF:	103,378	176	163	13	14.55	965	120,166

Time (sec.), solved instances, solved instances where a plan was found and not found (with length ≤ 200), average time ($\overline{t}$), number of backtracks ($\overline{b}$) and assignments ($\overline{a}$).

Experiments (2/3): Cactus Plot of Run Times



Experiments (3/3): Detailed Solving Statistics

Dungeon (81 solved planning instances)				
		DepQBF	incDepQBF	diff. (%)
Per instance	$\overline{a}$:	2,114,146	1,509,049	-28.6
	$\overline{b}$:	20,497	15,276	-25.4
	$\overline{t}$:	15.47	7.88	-49.0
	$\tilde{a}$:	1,388	1,391	+0.2
	$\tilde{b}$:	13	11	-15.3
	$\tilde{t}$:	1.01	0.37	-63.8

Average and median number of assignments ($\overline{a}$ and $\tilde{a}$, respectively), backtracks ($\overline{b}, \tilde{b}$), and workflow time ($\overline{t}, \tilde{t}$) for planning instances where both workflows using DepQBF and incDepQBF found the optimal plan.

Conclusions

Incremental QBF-Based Approach to Conformant Planning:

- Iterative stepwise refinement of plan length.
- Exploit information learned from previous steps.
- Precision: we always find the shortest plan (given sufficient time and memory).
- Certification that no plan of certain length exists.
- Incremental solving outperforms non-incremental solving in our tool.
- Incremental solving compares favourably to heuristic planning tools.

Future Work:

- Our tool currently supports preprocessing with non-incremental solving only.

Java Interface of DepQBF:

- DepQBF4J, comes with DepQBF: `http://lonsing.github.io/depqbf/`